

SEP 23 2003



PATENT

Customer No. 22,852

Attorney Docket No. 07781.0089-00000

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:)
)
Frank JENTSCH et al.) Group Art Unit: Not Yet Assigned
)
Application No. 10/612,011) Examiner: Not Yet Assigned
)
Filed: July 3, 2003)
)
For: DYNAMIC BREAKPOINTS FOR)
COMPUTER SOFTWARE)

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

CLAIM FOR PRIORITY

Sir:

Under the provisions of 35 U.S.C. § 119, Applicants hereby claim the benefit of the filing date of European Patent Application Number 02014790.6, filed July 4, 2002, for the above identified United States Patent Application.

In support of Applicants claim for priority, a certified copy of the priority application is filed herewith.

Respectfully submitted,

FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, L.L.P.

By: _____
C. Gregory Gramenopoulos
Reg. No. 36,532

Dated: September 23, 2003

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com



**Europäisches
Patentamt**

**Eur pean
Patent Office**

**Office européen
des brevets**

Bescheinigung

Certificate

Attestation

Die angehefteten Unterla-
gen stimmen mit der
ursprünglich eingereichten
Fassung der auf dem näch-
sten Blatt bezeichneten
europäischen Patentanmel-
dung überein.

The attached documents
are exact copies of the
European patent application
described on the following
page, as originally filed.

Les documents fixés à
cette attestation sont
conformes à la version
initialement déposée de
la demande de brevet
européen spécifiée à la
page suivante.

Patentanmeldung Nr. Patent application No. Demande de brevet n°

02014790.6

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk

DEN HAAG, DEN
THE HAGUE, 21/11/02
LA HAYE, LE



Europäisches
Patentamt

European
Patent Office

Office européen
des brevets

Blatt 2 der Bescheinigung
Sheet 2 of the certificate
Page 2 de l'attestation

Anmeldung Nr.:
Application no.:
Demande n°: 02014790.6

Anmeldetag:
Date of filing:
Date de dépôt: 04/07/02

Anmelder:
Applicant(s):
Demandeur(s):
SAP Aktiengesellschaft
69190 Walldorf
GERMANY

Bezeichnung der Erfindung:
Title of the invention:
Titre de l'invention:
Dynamic breakpoints for computer software

In Anspruch genommene Priorität(en) / Priority(ies) claimed / Priorité(s) revendiquée(s)

Staat:
State:
Pays:

Tag:
Date:
Date:

Aktenzeichen:
File no.
Numéro de dépôt:

Internationale Patentklassifikation:
International Patent classification:
Classification internationale des brevets:

/

Am Anmeldetag benannte Vertragsstaaten:
Contracting states designated at date of filing:
Etats contractants désignés lors du dépôt:

AT/BE/CH/CY/DE/DK/ES/FI/FR/GB/GR/IE/IT/LI/LU/MC/NL/PT/SE/TR

Bemerkungen:
Remarks:
Remarques:

- 1 -

2002-088-EP-

DYNAMIC BREAKPOINTS FOR COMPUTER SOFTWARE**Background of the Invention**Field of the Invention

The technical field of this invention is in the area of
5 electronic data processing. More particularly, the invention
relates to methods, computer program products and systems for
debugging a program code.

Description of the Related Art

When developing a computer program, debugging of the program
10 code is often a difficult and time consuming work. Debugging is
often performed in the following manner: finding the program
part which causes problems, finding the relevant lines of the
program code, setting of breakpoints at places in the program
code, which are of interest regarding the status of certain
15 variables, and executing the program code with a debugging
software (debugger). When executing the debugger, the run-time
environment of the program developing environment starts
executing the program code, stops its execution at the first
breakpoint in the order of the execution and usually displays
20 the content of variables for analysis. A failure in the program
code (bug) needs not necessarily be present. The debugger can
be used to only show the content of certain variables as well.

This procedure has certain disadvantages. The performance of
25 the first three steps mentioned above is a lot of workload if
the debugging has to be frequently repeated. Program developers
and coworkers, who have to maintain the program, need a good
understanding of the total program code, although only part of
the code might be interesting for the debugging purpose. An
30 other point is that the breakpoints are automatically deleted

- 2 -

2002-088-EP-

when closing the program. So, when starting the debugging software anew in a new session, the breakpoints have to be set again.

- 5 Thus, there is a need for a method and/or debugging system providing a more efficient solution of the problems described above.

Summary of the Invention

- 10 In accordance with the invention, as embodied and broadly described herein, methods and systems consistent with the principles of the invention provide for debugging computer program code by means of a debugging software, which provides software means for causing the debugging software to stop at a breakpoint set in said computer program code, comprising:
- 15 -- making the stop of the debugging software dependent on one or more predefinable conditions.

- 20 In accordance with another aspect, the invention, as embodied and broadly described herein, methods and systems consistent with the principles of the invention provide a computer system for debugging computer program code by means of a debugging software, which provides software means for causing the debugging software to stop at a breakpoint set in said computer program code comprising:
- 25 - a memory having program instructions;
- an input means for entering data;
- a storage means for storing data;
- a processor responsive to the program instructions to:
-- making the stop of the debugging software dependent on one
30 or more predefinable conditions.

The invention is further directed to a computer readable medium, a computer program and a carrier signal comprising

- 3 -

2002-088-EP-

instructions for debugging program code according to the method as described above.

Additional objects and advantages of the invention will be set forth in part in the description, or may be learned by practice of the invention. The objects and advantages of the invention will be realized and attained by means of the elements and combinations particularly pointed out in the appended claims. Preferred implementations of the invention are disclosed in the detailed description section and in the dependent claims.

It is understood that both the foregoing general description and the following detailed description are exemplary and explanatory only and are not restrictive of the invention, as claimed.

Brief Description of the Drawings

The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate implementations of the invention and, together with the description, explain the principles of the invention. In the drawings,

Fig. 1 is an exemplary computer system for implementing the features of the principles of the present invention;

Fig. 2 shows a schematic representation of an exemplary screen mask to activate or deactivate breakpoints.

Detailed description

Reference will now be made in detail to the principles of the invention, examples of which are illustrated in the accompanying drawings. Examples, mentioned therein, are

- 4 -

2002-088-EP-

intended to explain the invention and not to limit the invention in any kind.

Within the scope of this invention, the terms of the invention
5 used in this specification have the meaning that is usual in
programming languages like C, C++, JAVA, ABAP, ABAP Objects.
The following explanations of the invention are made with
reference to the figures 1 to 2. The given examples are of
explanatory nature only and are not intended to limit the scope
10 of the invention.

Particularly, a breakpoint or a breakpoint instruction, both
terms being used herein alternatively, is any software means,
which causes a debugging software to stop at that point of the
15 program code, at which the breakpoint or breakpoint instruction
is set. Debugging a program code means, within the scope of
this invention, to execute a program code by means of a
debugging software. Whether the program code contains failures
(bugs) or not is irrelevant. A breakpoint according to the
20 invention can be implemented in a program code by a macro call.
A macro thereby is a abbreviation of a program coding
containing the breakpoint instruction. In the macro, the
breakpoint is included in a conditional instruction having one
or more conditions, which must be fulfilled, if the breakpoint
25 instruction can be executed by the run-time environment. The
implementation by a macro is not inevitably necessary. However,
it reduces the programming effort considerably.

A condition (abbreviated as "cond") can be any type of logical
30 comparison of data known to the skilled artisan. Two or more
conditions can be combined by logical connections "AND", "OR"
and/or "XOR" as may be appropriate in the respective case.

The conditions can be structured such, that in all macros of a
35 given program an identical condition has to be fulfilled. In

- 5 -

2002-088-EP-

this case the identical condition works like a main switch. Further in different macros, additional to such a general condition, additional second, third, fourth or more conditions can be implemented. Structures mixed with "AND", "OR" and/or
5 "XOR" connections are also possible.

The macros can be stored in one or more separate files, which can be included into the program code by include instructions in the definition part of the program. This procedure makes the
10 breakpoints persistent with respect to the closing of the program.

Advantageously, different macros can be defined using different sets of conditions. The different sets of one or more
15 conditions can be stored in one or more arrays or in one or more tables in volatile and/or non volatile memory. Such an array or table can be made accessible for only one type of macro. The set or sets of conditions can be edited during the debugging procedure. This allows, by setting or deactivation
20 software switches, that all or only certain types of breakpoints, such, which have one common condition included in the respective macros, can be activated or deactivated by a single step. This can be implemented e.g. by a screen mask, in which a software switch can be activated or deactivated by a
25 mouse click or a key stroke. The screen mask according to this invention is a graphical user interface, by means of which reading and writing access is provided to the array, in which the one or more conditions are stored.

30 A further preferred implementation is a macro in which one of the conditions whether a string variable handed over by the macro is identical to a respective variable stored in the arrays or tables mentioned above. In this case, a query is started to check, whether a preset string is equal to a string
35 given in the macro.

- 6 -

2002-088-EP-

A preferred implementation of a breakpoint, for example, can be a macro, in which the one or more conditions are realized by the classical "if" instruction as pointed out in more detail in the following example. Alternatively, "case" or "select case" or similar instructions may be used as well.

```
MACRO_1:
IF cond0 = TRUE AND cond1 = TRUE THEN
10  BREAKPOINT
ENDIF
```

An example of the invention by using a breakpoint of the latter type and a screen mask is shown by the schematic drawings in figures 1 and 2:

Fig. 1 shows a computer system 1 for allowing a user to debug program code by means of a debugger according to the principles of the present invention. Computer system 1 comprises a server 20 3 with a central processing unit 5, a net connection 4 for connection with at least one input/output means 2 including screen and keyboard, at least one data storage means 7 and memory 6 including program code and debugging software. On the screen, a screen mask like a window 8 for providing read and write access to the storage means 7 is shown. In Fig. 2 the invention is described in more detail. In a program code 6 three types of macros including an if-instruction and a breakpoint instruction are inserted (breakpoint). The three types of breakpoints are named Macro_1, Macro_2 and Macro_30 Macro_ "XYZ". In each breakpoint, a different set of conditions has to be fulfilled, if the debugger shall be caused to stop at the respective breakpoint: In Macro_1 cond0 and cond1, in Macro_2 cond0 and cond2, and in Macro_3 cond0 and condn have to be fulfilled. Whether a condition is fulfilled or not can be 35 decided by the contents of the respective fields of a data

- 7 -

2002-088-EP-

array 11, where respective fields are allocated to respective conditions. In the example in Fig. 2, fields named cond*"i"* with *i* from 1 to *n* are allocated to the conditions of the same name. In the example cond0 reads as: if the content of field cond0 in the Data Array is TRUE (i.e. contains the boolean value TRUE), then cond0 is fulfilled. Cond*n* reads as: if the field cond*n* of the Data Array is equal to "XYZ" (i.e. contains the string "XYZ"), then cond*n* is fulfilled.

As indicated in Fig. 2, the fields of the data array can be accessed by read and write operations by a user via a graphical interface like a window 8 on a screen. A user can set a field in the data array to TRUE or FALSE (boolean variable) by clicking with a mouse on boxes 9 named according to the conditions they refer to. A "V" indicates that the content of the respective field in the data array 11 has the content TRUE. A blank indicates that the content of the respective field in the data array 11 has the content FALSE. The string for cond*n* can be edited via a keyboard in a screen field 10. The screen window 8 can have additional usual attributes like OK button to save in storage and Cancel to escape without save. As can be deduced from Fig. 1, cond0 is a general condition, which allows to completely deactivate all macros in the program by one click. In the example of Fig. 1, the breakpoint instruction only in Macro_2 and Macro_"XYZ" will be executed, what means that the debugger will stop at the respective places, where the macros are incorporated in the program code.

Instead of the boolean variables used in this example any other set of variables can be used, e.g. 0 and 1, A and B or mixtures thereof.

The invention is of a general nature and can be used debugging software in programming languages or rather programming environments, which support a statement to define a breakpoint, such as ABAP, ABAP Objects or Visual Basic. A technical

- 8 -

2002-088-EP-

advantage of the breakpoints according to the inventive concept is that such breakpoints, when set in a program code, are persistent even if the debugging session is closed.

- 5 Computer programs based on the written description of this invention are within the skill of an experienced developer.

The invention can be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. The invention can be implemented as a computer program product, i.e., a computer program tangibly embodied in an information carrier, e.g., in a machine-readable storage device or in a propagated signal, for execution by, or to control the operation of, data processing apparatus, e.g., a programmable processor, a computer, or multiple computers. A computer program can be written in any form of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program can be deployed to be executed on one computer or on multiple computers at one site or distributed across multiple sites and interconnected by a communication network.

- 25 Method steps of the invention can be performed by one or more programmable processors executing a computer program to perform functions of the invention by operating on input data and generating output.

- 30 Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read-only memory or a random access memory or both. The essential elements of a computer
- 35

- 9 -

2002-088-EP-

are a processor for executing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto-optical disks, or optical disks. Information carriers suitable for embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, such as EPROM, EEPROM, and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in, ASICs (application-specific integrated circuits).

To provide for interaction with a user, the invention can be implemented on a computer having a display device such as a CRT (cathode ray tube) or LCD (liquid crystal display) monitor for displaying information to the user and a keyboard and a pointing device such as a mouse or a trackball by which the user can provide input to the computer. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback, such as visual feedback, auditory feedback, or haptic feedback; and input from the user can be received in any form, including acoustic, speech, or haptic input.

Modifications and adaptations of the present invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein. The foregoing description of an implementation of the invention has been presented for purposes of illustration and description. It is not exhaustive and does not limit the invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may

- 10 -

2002-088-EP-

be acquired from the practicing of the invention. For example, the described implementation includes software, but systems and methods consistent with the present invention may be implemented as a combination of hardware and software or in hardware alone. Additionally, although aspects of the present invention are described for being stored in memory, one skilled in the art will appreciate that these aspects can also be stored on other types of computer-readable media, such as secondary storage devices, for example, hard disks, floppy disks, or CD-ROM; the Internet or other propagation medium; or other forms of RAM or ROM. It is intended that the specification and examples be considered as exemplary only, with a true scope and spirit of the invention being indicated by the following claims.

15

- 11 -

2002-088-EP-

Claims:

1. A method for debugging computer program code by means of a debugging software, which provides software means for causing the debugging software to stop at a breakpoint set
5 in said computer program code, comprising:
-- making the stop of the debugging software dependent on one or more predefinable conditions.
2. The method of claim 1, wherein:
-- the one or more predefinable conditions are different
10 for at least two breakpoints.
3. The method of claim 1 or 2, wherein
-- the one or more predefinable conditions are stored in a data array.
4. The method of one of claims 1 to 3, wherein
15 -- the one or more predefinable conditions are identical for a predefinable type of breakpoint.
5. The method of one of claims 1 to 3, wherein
-- the one or more predefinable conditions are stored in a data array, which is accessible for only one type of
20 breakpoints.
6. The method of one of claims 1 to 5, wherein
-- the one or more predefinable conditions are changeable during the debugging process.
7. The method one of claims 1 to 6, wherein
25 -- the one or more predefinable conditions are stored in a non volatile storage.
8. The method of one of claims 1 to 7, wherein
-- the setting of the breakpoint is achieved by setting a macro call, comprising said breakpoint.

- 12 -

2002-088-EP-

9. The method as claimed in any of claims 3 to 8,
-- the data array is editable by means of a screen mask.
10. The method as claimed in any of claims 3 to 9, wherein
-- the data array is a table.
- 5 11. The method as claimed in any of claims 3 to 10, wherein
- the data array is accessible for read and write
operations via a graphical user interface.
12. A computer system for debugging computer program code by
means of a debugging software, which provides software
10 means for causing the debugging software to stop at a
breakpoint set in said computer program code comprising:
- a memory having program instructions;
- an input means for entering data;
- a storage means for storing data;
15 - a processor responsive to the program instructions to:
-- making the stop of the debugging software dependent on
one or more predefinable conditions.
13. The computer system of claim 12, wherein
-- the one or more predefinable conditions are different
20 for at least two breakpoints.
14. The computer system of claim 12 or 13, wherein
-- the one or more predefinable conditions are stored in a
data array.
15. The computer system as claimed in any of claims 12 to 14,
25 wherein
-- the one or more predefinable conditions are identical
for a predefinable type of breakpoint.
16. The computer system as claimed in any of claims 12 to 15,
wherein
30 -- the one or more predefinable conditions are stored in a

- 13 -

2002-088-EP-

data array, which is accessible for only one type of breakpoints.

17. The computer system as claimed in any of claims 12 to 16, wherein

5 -- the one or more predefinable conditions are changeable during the debugging process.

18. The computer system as claimed in any of claims 12 to 17, wherein

10 -- the one or more predefinable conditions are stored in a non volatile storage.

19. The computer system as claimed in any of claims 12 to 18, wherein

-- the setting of the breakpoint is achieved by setting a macro call, comprising said breakpoint.

15 20. The computer system as claimed in any of claims 14 to 19 -- the data array is editable by means of a screen mask.

21. The computer system as claimed in any of claims 14 to 21, wherein

-- the data array is a table.

20 22. The computer system as claimed in any of claims 14 to 21, wherein

- the data array is accessible for read and write operations via a graphical user interface.

25 23. A computer readable medium comprising instructions for debugging computer program code by means of a debugging software, which provides software means for causing the

30 debugging software to stop at a breakpoint set in said computer program code, comprising instructions for allowing a user to perform the method of any of claims 1 to 11, if said instructions are executed on a computer.

- 14 -

2002-088-EP-

24. A computer program comprising program instructions for allowing a user to perform the method of any of claims 1 to 11, if said computer program is executed on a computer.

25. A computer data signal embodied in a carrier wave

5

comprising:

computer executable instructions which cause a computer to provide means for allowing a user to perform the method of any of claims 1 to 11.

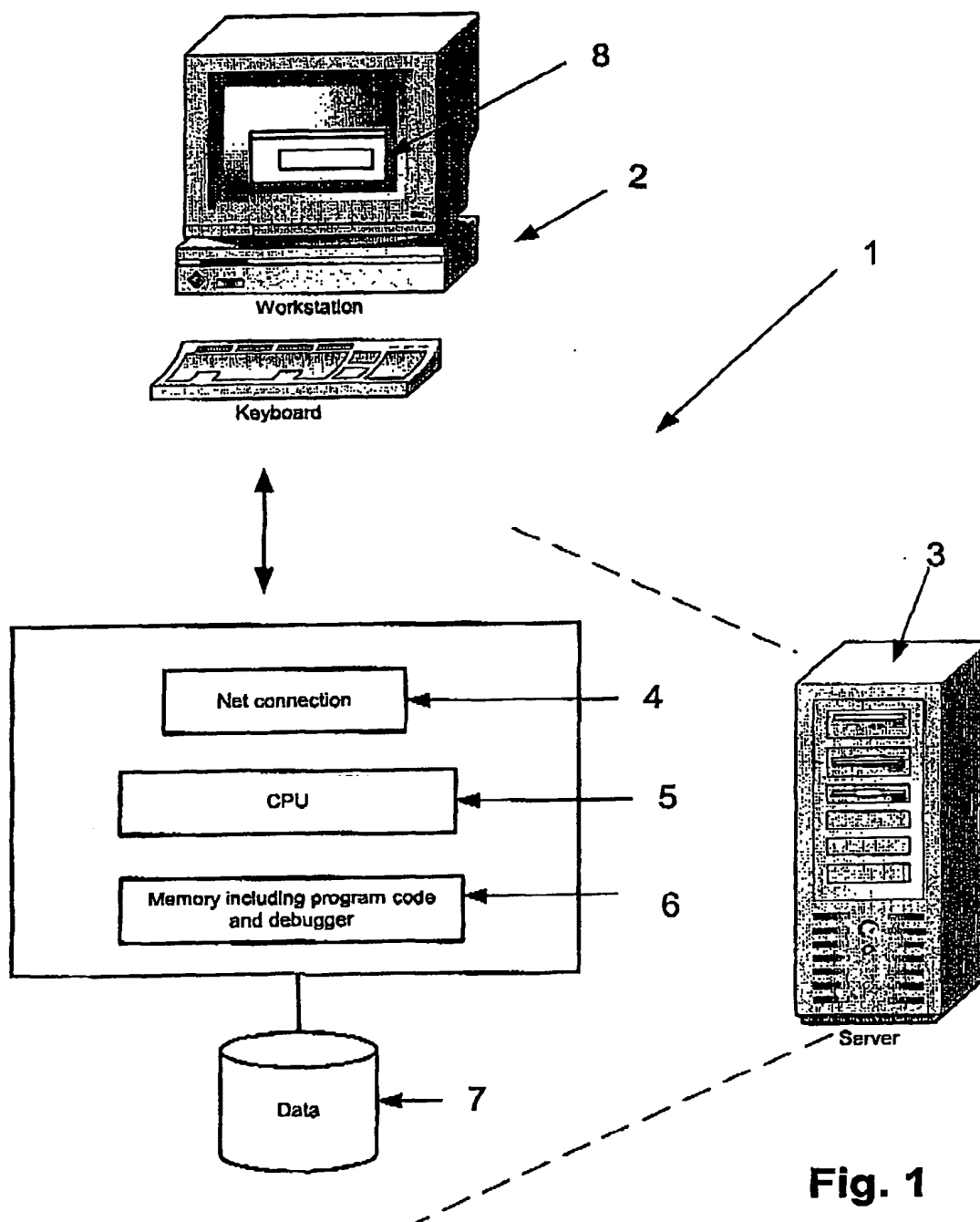
Abstract**5 DYNAMIC BREAKPOINTS FOR COMPUTER SOFTWARE**

The Invention relates to a process for debugging computer program code by means of a debugging software, which provides software means for causing the debugging software to stop at a breakpoint set in said computer program code, comprising:

- 10 -- making the stop of the debugging software dependent on one or more predefinable conditions.

(Fig. 2)

1/2



2/2

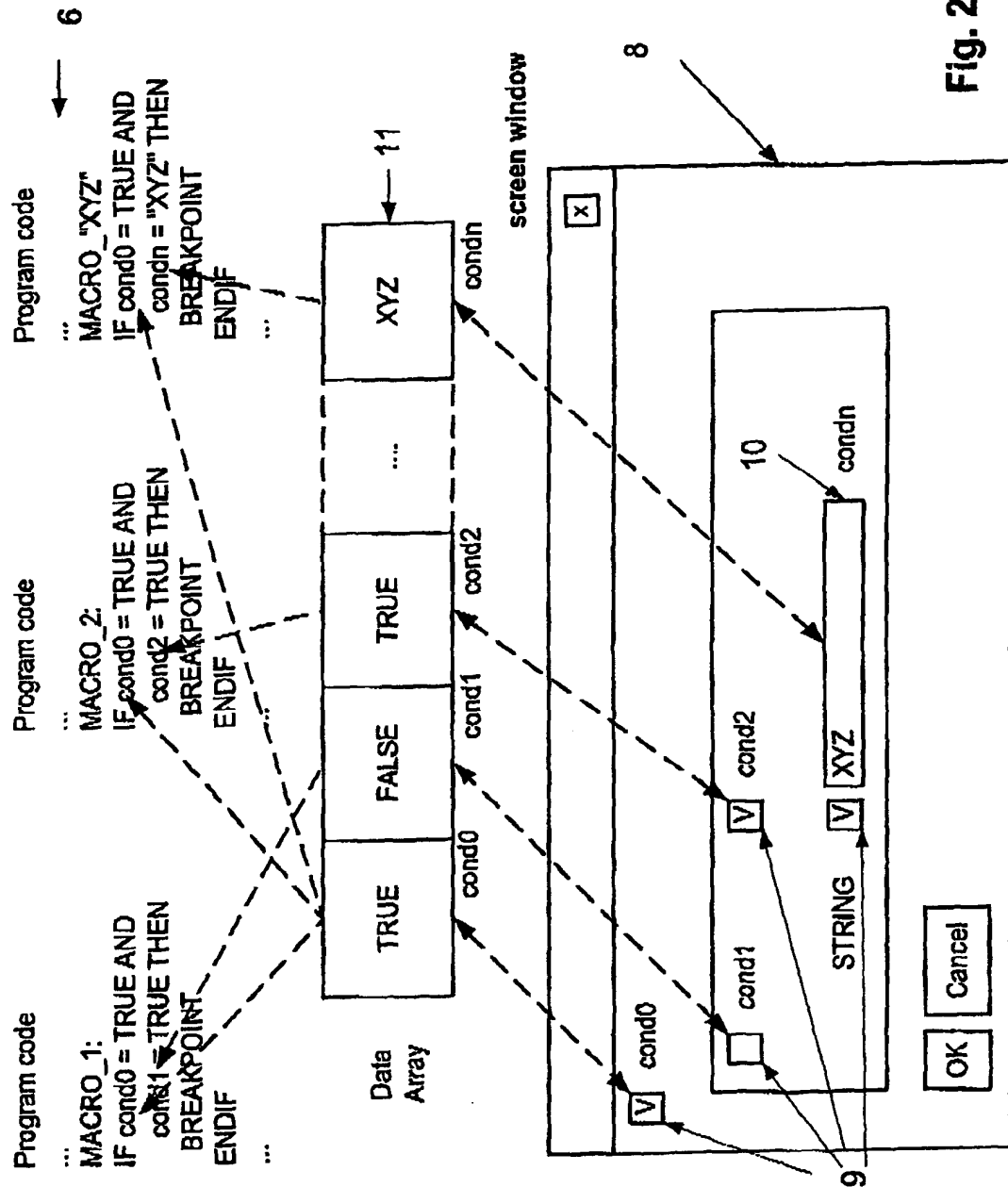


Fig. 2

